1 TITLE

2

3 Machine, process and manufacture for synchronizing data across integrated

4 applications

5

6 CLAIM OF PRIORITY/CROSS REFERENCE OF RELATED

7 APPLICATION(S)

8

9 Not applicable

10

11 STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR

12 DEVELOPMENT

13

14 Not applicable.

15

16 REFERENCE OF AN APPENDIX

17

18 Appendices A-B are contained herein. A portion of the disclosure of this patent

19 document may contain material, which is subject to copyright/trademark

20 protection. The copyright/trademark owner has no objection to the facsimile

21 reproduction by anyone of the patent document or the patent disclosure, as it

1   appears in the Patent and Trademark Office patent file or records, but otherwise

2   reserves all copyright/trademark rights whatsoever.

3

4

5   BACKGROUND

6

7   1. Field of the Invention

8

9   The present invention relates generally to data processing and more particularly,

10   to a novel machine, process and manufacture for synchronizing data across a

11   plurality of integrated applications.

12

13   2. Description of Related Art

14

15   Application integration is the process of bringing data or a function from one

16   application program together with that of another application program.

17   Implementing application integration has previously been a tedious process

18   involving long development and programming hours. However, the current trend

19   is to use specialized integration products (prepackaged "middleware" solutions),

20   such as message brokers and applications servers, to provide a common

21   connecting point among disparate applications.

1

2   Several patents and publications disclose various application integration methods,

3   portions of which are briefly summarized as follows:

4

5   U.S. Patent No. 6,236,994 entitled "Method and apparatus for the integration of

6   information and knowledge," issued on May 22, 2001 to Swartz, et al., and

7   discloses a method and apparatus for "integrating the operation of various

8   independent software applications directed to the management of information

9   within an enterprise. The system architecture is, however, an expandable

10   architecture, with built-in knowledge integration features that facilitate the

11   monitoring of information flow into, out of, and between the integrated

12   information management applications so as to assimilate knowledge information

13   and facilitate the control of such information. Also included are additional tools

14   which, using the knowledge information enable the more efficient use of the

15   knowledge within an enterprise, including the ability to develop a context for and

16   visualization of such knowledge."

17

18   U.S. Patent No. 6,256,676 entitled, "Agent-adapter architecture for use in

19   enterprise application integration systems," issued on July 3, 2001 to Taylor, et

20   al., and discloses "an agent-adapter architecture used in systems and methods to

21   integrate applications of the type normally deployed across a networked

1   enterprise. A plurality of adapters, each of which is adapted to perform a discrete

2   function associated with respective ones of the plurality of enterprise applications

3   is encapsulated by an agent. The agent is extensible, including one or more

4   embedded objects, each of which is adapted to perform a discrete function that

5   may or may not be associated with respective ones of the plurality of enterprise

6   applications."

7

8   Enterprise Application Integration, A Wiley Tech Brief by Willam A. Ruh et al,

9   Wiley Computer Publishing, 2001, describes various technologies, architectures

10 and approaches currently available for application integration.

11

12 Finally several integrated-related Internet resources such as the "EAI Journal,"

13 www.eaiijournal.com and the "EAI Forum," www.eaiforum.com, describe the

14 current state of application integration technologies.

15

16

17 SUMMARY OF THE INVENTION

18

19 One of several objects of the present invention (sometimes referred to as PDX) is

20 to provide user-driven, on-demand integration of applications, particularly

21 primarily stand-alone applications.

1

2   Further objects of the present invention include, but are not limited to:

3   1) providing a link to a "vertical" integration mechanism to enable the

4   horizontally integrated applications to integrate with other platform resources,

5   such as mainframes and servers (Unix and NT), 2) streamlining workflows,

6   3) eliminating redundant data, 4) move data among integrated applications with

7   minimal effort, 5) linking data records and synchronizing linked data records

8   across applications, 6) providing a migration path to a future state, and

9   7) minimizing data required by applications.

10

11   Therefore in accordance with one aspect of the present invention, there is

12   generally provided an apparatus, method and article of manufacture for

13   integrating a plurality of heterogeneous applications using a common integration

14   architecture wherein said apparatus, method and article of manufacture employs a

15   Links Table for associating related data. Utilization of the Links Table enhances

16   processing time over those techniques that search cumbersome data stores of

17   integrated applications for relevant information during synchronization.

18

19   The above-mentioned aspect(s) and other aspects, objects, features and

20   advantages of the present invention will become better understood with regard to

21   the following description, appended claims, and accompanying drawings.

1

2

3   BRIEF DESCRIPTION OF THE DRAWING(S)

4

5   Referring briefly to the drawings, embodiments of the present invention will be

6   described with reference to the accompanying drawings in which:

7

8   FIG. 1 is a general representation of various components that comprise an

9   integration architecture constructed in accordance with the teachings herein;

10

11   FIG. 2 depicts an exemplary message structure in accordance with the teachings

12   herein;

13

14   FIG. 3 depicts an exemplary user interface in accordance with the teachings

15   herein;

16

17   FIG. 4 depicts an exemplary synchronization flow in accordance with the

18   teachings herein;

19

20   FIGS. 5-10 each depict a detail of the flow set forth in FIG. 4;

21

1   FIG. 11 depicts a Links Table in accordance with the teachings herein.

2

3   FIGS. 12-16 depict exemplary application flows in accordance with the teachings

4   herein.

5

6   FIGS. 17-22 are representations of user interface screens depicting aspects of the

7   present invention.

8

9

10   DETAILED DESCRIPTION

11

12   Referring more specifically to the drawings, for illustrative purposes the present

13   invention is embodied in the system configuration, method of operation and

14   product or computer-readable medium, such as floppy disks, conventional hard

15   disks, CD-ROMS, Flash ROMS, nonvolatile ROM, RAM and any other

16   equivalent computer memory device, generally shown in Figures 1 – 22. It will

17   be appreciated that the system, method of operation and product may vary as to

18   the details of its configuration and operation without departing from the basic

19   concepts disclosed herein.

20

21   GLOSSARY

1

2   In describing the present invention, the following terms are used herein.

3

4   "Data store" is a place where information is saved, preferably, in a persistent

5   manner (e.g. on a hard drive). It may include relational databases, flat files, and

6   proprietary storage formats.

7

8   "Horizontal integration" is integration across a single platform as opposed to

9   integration between different platforms (e.g. client and server).

10

11   Integration software refers to the software/components used to synchronize

12   information between applications.

13

14   "IOD" or "Integration on demand" is a user-driven approach to integration and

15   not an automated replication model.

16

17   Vertical integration is integration between two or more platforms.

18

19   Working Client is that person whose information is designated as the current

20   working set for any particular application and may not necessarily be a "client" of

21   the enterprise as that term is used herein.

1

2

3   INTEGRATED ARCHITECTURE

4

5   To facilitate the integration and synchronization of required information, aspects

6   and features of the present invention are embodied in a common integrated

7   architecture. FIG 1, illustrates on example of such an integrated architecture 100

8   constructed in accordance with the teachings presented herein. As shown, the

9   integrated architecture comprises several interrelated components, namely an

10   Integration Engine having an Integration Engine Service Adapter and an

11   Integration Engine Data Store associated therewith (collectively enumerated as

12   105), a plurality of Applications having associated Application Service Adapters

13   and Application Data Stores (collectively enumerated as 110), Messages 115

14   having a predefined syntax, and a Dashboard user interface 120, all arranged in a

15   logical hub-and-spoke configuration.

16

17   Together the Integration Engine, its Service Adapter and Data Store, function as

18   the "hub" of the architecture. Responsibilities of the integration engine include

19   routing messages between service adapters based on type or content, transforming

20   a message or message content based on the requirements of the integrated

21   applications, and controlling the flow of information between service adapters.

1

2   The predefined Messages form the spokes. The content of every Message

3   conforms to a standard syntax. All applications/resources produce and consume

4   Messages that conform to a standard syntax, thus the present solution supports

5   "plug-and-play" capabilities.

6

7   Finally, the Applications and the Application Service Adapters are the ends of the

8   spokes. Depending on a particular task, Application Service Adapters can either

9   serve as sources or as destinations and are responsible for accessing

10   applications/resources to retrieve requested information and transforming this

11   information into a common syntax and back again to its original format.

12

13   Attention now turns to details of the aforementioned components.

14

15

16   INTEGRATION ENGINE

17

18   The Integration Engine, its Service Adapter and Data Store are situated at center

19   of the architecture. Together, these integration components implement intelligent

20   messaging by triggering and executing integration flows to process events and

21   rules that evaluate, modify, and route event data. Specific responsibilities include

1   setting application definitions for integrated applications, setting Dashboard's

2   settings, and implementing updates and/or additions to the Links Table (see

3   below).

4

5   The Integration Engine's Service Adapter can be called directly from the

6   Dashboard or the integration flows.

7

8   The design specification for the Integration Engine Service Adapter is set forth as

9   Appendix B.

10

11

12   APPLICATION SERVICE ADAPTERS

13

14   An adapter is an access point (logic) logic that provides access to the application

15   in a structured manner.   Thus, an adapter is an interface into the application that

16   defines the requests the receiver will accept while hiding the underlying

17   complexity of accomplishing the integration.

18

19   The Application Service Adapters herein are built to be plug and play with the

20   system. That is to say, a new Application Service Adapter can be plugged in and

1   removed from the architecture without impacting the remaining Application

2   Service Adapters.

3

4   Each application has its own data requirements. Typically, data requirements will

5   not match from application to application. Therefore, it is the responsibility of the

6   Application Service Adapter to understand and provide services to its underlying

7   data store and further perform the necessary business logic to the data being

8   passed to or retrieved from it.

9

10   While all Application Service Adapters speak in a standard syntax, nonetheless

11   should an application require another standard, it can easily be supported by the

12   transformation capabilities of the Integration Engine. To that end, the Integration

13   Engine communicates with the Application Service Adapters via predefined

14   Messages (see below).

15

16   The design specification for an Application Service Adapter is set forth as

17   Appendix C.

1   MESSAGES

2

3   The predefined Messages recognized by the present invention form the spokes in

4   the integrated architecture. In FIG 3, for visual simplicity, the spokes also include

5   the technology transport of the messages. The content of every Message conforms

6   to a standard syntax.   Specifically, the structure of the Messages created and

7   processed in the present invention may be logically divided into three main

8   sections, a Message Root section 202, a Message Envelope section 204 and a

9   Message Body section 206. The Message Envelope section is further divided into

10   a Source section 208 and a Destination section 210. The Message Body section

11   is further divided a Parameters section 212 and a Payload section 214. The

12   Payload section is still further divided into a Status section 216, a Links section

13   218, and a Pay Load Item section 220. Each of the foregoing sections and

14   subsections will now be further explained below.

15

16   1. Message Root

17

18   The Message Root section contains header information about a given message.

19   The Message Root comprises an IONS identifier (IONSID) field and a message

1   request type (RequestType) field. A description of each of the foregoing fields

2   follow.

3

4	Property	Description
5	IONSID	A unique identifier for a user, e.g., the
6		user's Windows® operating system login
7		id.
8		
9	RequestType	The name/type of the request message, e.g.,
10		GetPerson_RQ message or SyncPerson_RQ
11		message. See Appendix A for additional message
12		types.

13

14   2. Message Envelope

15

16   The Message Envelope section comprises a Source section and a Destination

17   section containing routing information, which lets the Integration Engine know

18   which Service Adapter(s) to send the message to. In cases where source

1 information is not needed, the initiator of the message need not supply source

2 information.

3

4 Note: in order to maintain the principle that a user should have exactly the same

5 rights to the information in an application data store that they have when using the

6 application itself, we must maintain information about users.

7

8 A. Source

9

10 The Source section consists of several fields having information pertaining to the

11 Service Adapter of the Source Application. The relevant fields are described

12 below.

13

14 Property	Description
15	
16 DataSourceDescription	A description of the data source, usually the
17	ODBC DSN or a File Name.
18	
19 Name	The name of the Service Adapter.
20	

1	UserID	An application user
2		identifier. Note: This is not the IONSID
3		mentioned above.
4		
5	Password	An application user password associated
6		with the UserID. The password is used to
7		validate an applications settings when a user
8		customizes the Dashboard. There is no
9		requirement to store the password.
10		
11	B. Destination	
12		

13 The Destination section consists of several fields having information pertaining to

14 the Service Adapter of the Destination Application. The relevant fields are

15 similar to that of the Source section and are described below.

16

17	Property	Description
18		
19	DataSourceDescription	A description of the data source,
20		usually the ODBC DSN or a File Name.
21		

1	Name	The name of the Service Adapter.
2		
3	UserID	An application user
4		identifier. Note: This is not the IONSID
5		mentioned above.
6		
7	Password	An application user password associated
8		with the UserID. The password is used to
9		validate an applications settings when a user
10		customizes the Dashboard. There is no
11		requirement to store the password.
12		
13		

14   3. Message Body

15

16   The Message Body section contains information to enable a receiver of the

17   message to process a request and further holds the requested information or data.

18   As described earlier, the Message Body is divided into two sections - Parameters

19   and Payload. A description the these sections follows.

20
21

1   A. Parameters

2

3   The Parameters section contains the parameters that a message requires. In cases

4   where a message does not utilize a parameter this section will be blank. The

5   Parameters section comprises two fields, which are described below.

6

7   Property   Description

8   Name   The name of the parameter

9   Value   The value of the parameter.

10

11   B. Payload

12

13   The Payload section contains the results of the message request. The Payload

14   section divides into three sub-sections, namely a Status section, a Links section

15   and a Payload Items section. The foregoing sub-sections are described below.

16

17   i. Status

18

19   The Status section contains information related to the completion of the message.

20   If the message is successful, it will contain a status code and description

21   indicating success. It is also here that you will find information about any errors

1   that were encountered during the messages execution. There can be many

2   occurrences of this section. The Status section comprises several fields, each of

3   which are described below.

4

5   <u>Property</u>   <u>Description</u>

6

7   StatusCode   The status code of the error.

8

9   Description   A description of the error.

10

11   OriginatedFrom   The name of the dynamic link library (DLL) that the error

12   occurred in.

13

14   ModuleName   The name of the module that the error occurred in.

15

16   MethodName   The name of the method where the error occurred.

17

18

19   ii. Links

20

1   The Links section is utilized during synchronization. Among other information,

2   the Links section contains a record of a Service Adapter's actions, that is whether

3   an "Add" or "Update" was done. In addition, it contains certain information a

4   Service Adapter needs during processing, for example, the unique identifiers

5   assigned to the Source and Destination Applications.   The Links section

6   comprises several fields, each of which are described below.

7

8   Property   Description

9

10   DataSourceDescription   A description of a data source, usually the

11   ODBC DSN or a File Name

12

13   Name   The Service Adapter's name

14

15   *SourceID   A unique identifier for the object, e.g. a

16   person or an organization, in the

17   Source Application

18

19   *DestinationID   A unique identifier for the object in the

20   Destination Application.

21

1	*PartyID	A unique identifier for the object within
2		the system.
3		
4	*ObjectType	Identifies the object as a person or an
5		organization.
6		
7	*ActionPerformed	The action that the Service Adapter
8		performed on the object, e.g., "Add" or
9		"Update"

10

11   * These properties can occur multiple times.

12

13   iii. Payload Item

14

15   The Payload Item section contains the results of a request. For example, a

16   receiver of a Search_RQ request message would place the search results in the

17   Payload Item section of a response message. There can be one or more instances

18   of a payload item. Since a request can be sent to multiple destinations, to track

19   what part, or "item", in the payload came from a particular Service Adapter the

20   following properties/fields are included at the beginning of each Payload Item

21   section.

	Property	Description
1		
2	Property	Description
3		
4	DataSourceDescription	The description of a data source. Usually the
5		ODBC DSN or a File Name.
6		
7	Name	The Name of the Service Adapter
8		

9   The rest of Payload Item section varies based on the request. Specific message

10   types used herein are set forth in the Appendices.

11

12

13   USER INTERFACE

14

15   The present system includes a graphical user interface that enables a user to work

16   with the aforementioned integration flows.

17

18   Accordingly, FIG. 3 depicts an exemplary embodiment of an "Integrated Client

19   Dashboard" graphical user interface ("Dashboard") utilized in the present

20   invention. The Dashboard is designed to facilitate user-driven data integration

21   across integration Applications via a flexible application workflow model.

1

2   For example and referring to the insurance industry, a user who starts the sales

3   process by entering information in a prospecting application (CDS) and then

4   moves on to discovery and analysis using a discovery application (DIS) and an

5   analysis application (PAS) can use the Dashboard to move client information

6   from the CDS to DIS/PAS applications by activating the buttons on the

7   Dashboard. Similarly, an alternative workflow is also supported wherein a user

8   begins prospecting by enter information into illustrations application (ISP) and

9   then moves to the discovery application.

10

11   As shown, the look and feel of the illustrated embodiment of the Dashboard is

12   based on the look and feel of the popular Shortcut Bar used in Microsoft® Office

13   suite. The Dashboard spans the length of the display device and is initially

14   situated at the bottom of the display medium, such as a window object or

15   computer screen. However, as is conventional, the Dashboard may be resized and

16   also positioned elsewhere on the display device as preferences dictates.

17

18   The Dashboard includes several areas, namely a Menu Access area 302, an

19   Integration area 304, a Non-Integrated area 306, and a Status area 308, which

20   together invoke aspects and features of the present invention. The details of each

21   of these areas will now be discussed.

1

2 Menu access area

3

4 The Menu access area comprises a Menu Access button. Upon selection of the

5 Menu access button a menu bar appears like that shown in FIG. 22. As indicated,

6 the menu bar provides access to certain commands/functions, for example, Auto

7 Hide, Customize, Help and Exit, that control certain aspects of the instant

8 invention. The Auto Hide command enables the Dashboard to reside behind other

9 applications displayed on the display device. The Customize command allows the

10 user to alter settings for the Dashboard, for example, adding/removing application

11 files to/from the Dashboard, modifying information about a particular application,

12 and modifying other attributes of the Dashboard, including color and size.

13

14 Selecting the Help command launches the Help facility and selecting the Exit

15 command exits the Dashboard. Other commands/function may also be displayed.

16

17

18 Integration area

19

20 The Integration area is a collection of buttons that serve as shortcuts for executing

21 and controlling the synchronization process describe herein.

1

2 For example, the embodiment shown in FIG. 3 depicts, a Help button (A), a

3 Search button (B), a MultiSync button (C) and several buttons representing

4 applications integrated with the present system (D-H).

5

6 Selecting the Search button launches a search applet for searching integrated

7 applications. The search applet returns and displays results based on the search

8 criteria entered. Using the returned results a user may take the following actions:

9 1) select a Working Client causing information about the Working Client to

10 appear in Status area of the Dashboard); 2) select the Working Client and view all

11 data relating to the Working Client, and 3) select a Working Client and launch

12 directly to the application where the client resides.

13

14 Selecting the MultiSynch button synchronizes data from Source to Destination

15 Applications.

16
17 Selecting any one of the integrated application buttons will process the Working

18 Client's information in accordance with the teachings expressed herein, for

19 example, push data from a source to the selected integrated application.

20

21 Non-Integrated area

1

2   The non-integrated area includes a non-integrated application button and one or

3   more specific application/website buttons. Upon selection of the non-integrated

4   application button, a drop down list appears. The drop down list sets forth at least

5   all external, non-integrated applications/web sites having buttons appearing on the

6   Dashboard. Selecting any one of the application/website buttons will launch the

7   particular application/website.

8

9   Status area

10

11   The Status area displays information that is relevant to the current Working

12   Client, for example, the name of and the Working Application containing the

13   Working Client.

14

15   When the Dashboard restarts, it will retain all prior settings at shut down

16   including the Working Client and Working Application information and the

17   Dashboard's last position on the display device

18

19   Other features/embodiments of the Dashboard

20

1   The Dashboard, in its most basic embodiment, may be customized to only consist

2   of a Menu Access area, an Integration area and a Status Area and yet still retain

3   the desired functionality.

4

5   In alternative embodiments of the Dashboard, certain areas of the Dashboard will

6   display hover text indicating the function of the area. For example, to display the

7   function of the Multi Sync button, a user positions the mouse over the button for a

8   few moments to generate a hover text stating "Send Working Client to Multiple

9   Applications."

10

11   Further, when a user right-mouse-button clicks anywhere on the Dashboard, the

12   system displays a shorter, modified menu bar. This menu has three of the same

13   functions as the regular menu – Auto Hide, Customize, and Exit. These behave

14   exactly the same as on the regular menu.

15

16

17   DATA SYNCHRONIZATION

18

19   An application of the present solution will now be described with reference to

20   FIGS. 4-11. The sections that follow demonstrate how customer demographic

21   information is modified across several integrated heterogeneous applications.

1

2   FIG. 4 depicts a high level view of an exemplary data synchronization flow in

3   accordance with the principles expressed herein comprising several interrelated

4   software modules, namely, a Set Working Client & Application module 400, a

5   Verify Links module 402, a Find Matches module 404, a Verify Destination

6   Application Availability module 406 and a Synchronization module 408.

7

8   When a user clicks on an application icon on the Dashboard signaling data

9   synchronization, an integration request message is generated by the dashboard

10   and sent to the Integration Engine where certain pre-processing steps are first

11   performed before the integration request message is handled. An integration

12   request message is a template by which all the messages described herein are

13   derived from. The messages have a certain attributes that are the same regardless

14   of what type of request is being made (for example, whether the request is a

15   Search, Sync, etc.). Specifically, all messages have the following properties,

16   IONSID, REQUEST_TYPE, SOURCE, DESTINATION, PARAMETERS.

17

18   As shown in FIG. 4, to begin synchronization, the Set Working Client &

19   Application module executes. This module performs the pre-processing steps of

20   verifying whether a desired Working Client has been properly selected and

21   whether a desired Working Application (Source Application) is available for

1   synchronization. If true, the Verify Links module 4 is dynamically created (see

2   below) and utilized as a basis for the synchronization flow.

3

4   If the Working Client is properly set and the Source Application is available for

5   synchronization, the Verify Links module 402 executes. This module is

6   dynamically created and utilized as a basis for the synchronization flow.

7   Specifically, this module first verifies that signatures exist in the Destination

8   Application data store. For example, when a signature check is performed, a

9   getSignature message is constructed and sent to a Destination Application's

10   Service Adapter. Next, an attempt to retrieve the links for the Working Client in

11   both the Source and Destination Applications is performed against the Integration

12   Engine's database.

13

14   Attention will now turn to the Links Table as that data structure is used herein. A

15   dynamic Link Table like that shown in FIG. 11 is populated whenever a user

16   establishes links among integrated applications for a particular

17   customer/person/client. As shown, the Link Table has five columns entitled, Link

18   Key, User Application Id, Party Id, Client Id and Last Sync Date.

19

1   The Link Key column contains unique identifiers associated with each row in the

2   Link Table. In the present example, each row is number sequentially. Thus, row

3   one has a link key of 1, row 2 has a link key of 2 and so on.

4

5   The User Application Id column contains unique identifiers associated with the

6   integrated applications. Thus, in the example illustrated in FIG 11, the CDS

7   application is assigned a user application id of 5, the ISP application is assigned a

8   user application id of 6 and the PAS application is assigned a user application id

9   of 7.

10

11   The Party Id column contains global identifiers associated with each

12   customer/person/client in the Integration Engine data store. Thus, in the example

13   illustrated in FIG. 11, customer/client/person William Brown is assigned a party

14   id of 2 and customer/client/person J. Doe is assigned a party id of 2. Notably, a

15   glance down the Party Id column immediately tells a reviewer that only two

16   people are currently linked in the Integration Engine data store.

17

18   The Client Id column contains unique identifiers associated with

19   customers/clients/persons in their respective native applications. Thus, in the

20   example illustrated in FIG. 11, the CDS application assigns

21   customer/client/person W. Brown a client id of 20 and assigns

1   customer/client/person J. Doe a client id of 200. The PAS application assigns a

2   client id of 20 to customer/client/person W. Brown and assigns

3   customer/client/person J. Doe a client id of 200.

4

5   The Last Sync Date column sets forth the most recent date synchronization was

6   done for a particular customer/clients/person. Thus, in the example illustrated in

7   FIG. 11, data was last synchronized in the CDS application for

8   customer/client/person W. Brown on 1/18/01 and in the ISP and PAS applications

9   on 5/22/01. For customer/client/person J. Doe, data was last synchronized in the

10   CDS and PAS applications on 2/4/01.

11

12   Due to the inherent nature of technology, unforeseen glitches may occur and as a

13   result cause an application's data to become corrupted. However, once an

14   application's data is restored, all of the unique identifiers in the application's data

15   store will change where the identifier is a sequentially generated one.

16   Consequently, the identifiers will no longer match what is stored in the Link table

17   for that application data store.

18

19   Without correction, a synchronize action will associate and overlay information

20   with the wrong individual/object/item in the application's data store.

21

1   Because of the necessity to provide correct and consistent information, the present

2   invention provides for the Links Table to be updated whenever a particular

3   application's data store has been corrupted, reloaded or refreshed.

4

5   The first time an application's data store is used, a signature record should be

6   written into the data store where the data store uses a unique identifier for an

7   individual/item/object that will change upon a data reload (e.g. have a

8   sequentially generated unique identifier).

9

10   The unique identifier for the signature record is recorded in the link table.

11   On start up, a check of all signature records will be performed. If a signature

12   record does not exist in one or more application data stores, an error message is

13   generated display an information and warning message for each application such

14   as "XYZ data appears to have been refreshed. Links are no longer valid. Do you

15   want to clear all links for this application?" If a user selects "Yes", the link table

16   entries for that application are reset and a new signature record is written to the

17   application data store. If "No" is instead selected, all add and update actions for

18   that application is disabled.   For example and referring to FIG 11, if the CDS

19   application's data store was corrupted and subsequently refreshed and a user

20   selects Yes, all links for CDS in the Links Table, that is Rows 1 and 4, would be

1   removed and replaced with updated information. If a user selects No the user will

2   unable to use the CDS application button on the Dashboard for synchronization.

3

4

5   Referring back to FIG. 4, if no links for the Working Client in the Source and

6   Destination Applications were found in the Verify Links module 402, the Find

7   Matches module 404 executes. Otherwise, the Find a Match module 404 is

8   bypassed and control passes to the Verify Destination Application Availability

9   module 406. The Find Matches module 404 searches within data stores

10 associated with selected Destination Applications to locate information matching

11 that of the Working Client.

12

13 Next, the Verify Destination Application Availability module 406 executes. This

14 module determines whether the desired Destination Application(s) is/are currently

15 available for synchronization. If true, a SyncPerson_RQ message will be created

16 (see below) and utilized in the Synchronization module described below.

17

18 Finally, the Synchronization module 408 executes. This module, among other

19 things, performs the desired task of synchronizing data from the Source

20 Application to the desired Destination Application(s).

21

1   For a better understanding of the present solution, the above modules will now be

2   further described in the sections that follow.

3

4

5   Set Working Client and Application

6

7   Referring to FIG 5, there is shown an exemplary block diagram detailing an

8   exemplary process flow of the Verification module in accordance with the

9   principles expressed herein.

10

11   In response to an integration request message, a determination is first made as to

12   whether the desired Working Client is set. If the Working Client is not set,

13   execution terminates and in one embodiment of the present solution, an error

14   message is generated indicating that the Working Client is not set. If, on the other

15   hand, the Working Client is set than a determination is made as to whether a

16   selected Working Application is available for data synchronization. During the

17   synchronization process, the Working Application will serve as the Source

18   Application. If the Source Application is not available for data synchronization,

19   execution terminates and in one embodiment of the present solution, an error

20   message is generated indicating that the Working Application is not available for

21   synchronization. However, if the Working Application is available for

1 synchronization, the Verify Links module will be dynamically constructed based

2 upon the original integration request message.

3

4 That is to say, in preparing the Verify Links module, certain information is

5 extracted from the original integration request message and included in the Verify

6 Links module, but not limited to, the Working/Source Application, the Working

7 Client, and all Destination Applications.

8

9 Finally, after the Working and Client applications have been properly verified and

10 the Verify Links module constructed, control passes to the next module in

11 sequence.

12

13 Verify Links

14

15 Upon completion of the Verification module, the Verify Links module executes in

16 accordance with the following exemplary process flow.

17

18 In one embodiment of the present invention, the Verify Links module logically

19 divides into two sub-processes. First, signatures are checked and second, links

20 between the Working Client and Destination Applications(s) are checked in

21 accordance with the following exemplary process flow.

1

2   A. Check Signatures in Destination Application's Data Store

3

4   Referring to FIG. 6, first, the Integration Engine's Service Adapter constructs a

5   GetSignature_RQ message based upon the contents of the VerifyLinks message.

6   Specifically, the Integration Engine's Service Adapter constructs a

7   GetSignature_RQ message for each application contained in the Source and

8   Destination sections of the VerifyLinks message envelope and transmits the same

9   to the Service Adapters of the Destination Application(s).

10

11   Next, in response to the GetSignature_RQ message, each Destination

12   Application's Service Adapter determines whether a signature already exists in

13   the Destination Application's data store and whether the Destination

14   Application's Service Adapter supports the integration request message. In

15   determining whether a signature exists, the Service Adapter checks the value in

16   the Status Code section of the GetSignature_RQ message. If a signature does not

17   exist, then the Destination Application's Service Adapter constructs a

18   ClearLinks_RQ message for the Destination Application and execution control

19   returns to process the next Destination Application contained in the

20   GetSignature_RQ message. If the Service Adapter supports the integration request

21   message, the Destination Application's Service Adapter constructs a

1   ClearLinksByDate_RQ message for the Destination Application and execution

2   control returns to process the next Destination Application contained in the

3   GetSignature_RQ message.

4

5   If the Destination Application's Service Adapter supports the integration request

6   message but the relevant signature does not exist in the Destination Application's

7   data store, both the ClearLinks_RQ and the ClearLinksByDate_RQ messages are

8   transmitted to the Integration Engine's Service Adapter. In response to the two

9   messages, the Integration Engine's Service Adapter constructs an

10 AddSignature_RQ message and transmits the message to the Destination

11 Application contained in the ClearLinks_RQ message. In response to the

12 AddSignature_RQ message, the Destination Application's Service Adapter will

13 add the signature to the Destination Application's data store. The foregoing

14 process is done for all Destination Applications wherein the Destination

15 Application's Service Adapter supports the integration request message but the

16 relevant signature does not exist in the Destination Application's data store.

17

18 B. Check Links using Links Table

19

20 Referring to FIG. 7, after signatures have been checked, links between the

21 Working Client and Destination Applications(s) are also checked.

1

2   First, the Integration Engine's Service Adapter constructs a CheckLinks_RQ

3   message based upon information derived from the original integration request

4   message and transmits the same to the Destination Applications. The

5   CheckLinks_RQ message will check to see if a link or links for the Working

6   Client/Person exist between the Source Application and the Destination

7   Application.

8

9   In response to the CheckLinks_RQ message and for each Destination Application

10   contained in the message envelope of the CheckLinks_RQ message, the

11   Integration Engine's Service Adapter will check the Links Table to determine

12   whether a link for the Working Client/Person exists between the Source

13   Application and the Destination Application. A CheckLinksResponse message is

14   then constructed and transmitted indicating the results of the query.

15

16

17   Find Matches

18

19   Referring to FIG. 8, the Find Matches module executes, if required, in accordance

20   with the following exemplary process flow. Note the Find Matches module

21   executes only if no link was found in the previous module.

1

2   For each Destination Application contained in the message envelope of the

3   original integration message, the following steps occur:

4

5   First, the Integration Engine Service Adapter constructs and transmits a

6   ServiceAvailable_RQ message to the Destination Application's Service Adapter

7   to determine whether the Destination Application is available for synchronization

8   and more particularly, whether the Destination Application's Services are

9   available.

10

11   If the response to the ServiceAvailable_RQ message is positive (that is the

12   Destination Application has synchronization capability), The Integration Engine

13   Service Adapter issues a GetPersonDetails_RQ message to the Source

14   Application's Service Adapter. In response to GetPersonDetails_RQ message,

15   the the Source Application's Service Adapter retrieves and transmits the desired

16   customer demographic information. If, however, the Destination Application's

17   Services are not available, execution terminates and, in one embodiment of the

18   present solution, an error message is generated indicating that the Destination

19   Application is not configured for synchronization.

20

1   Next, a determination is made as to whether a link exists for the Working Client

2   between the Source Application and the Destination Application.

3

4   If a link exists for the Working Client between the Source Application and the

5   Destination Application, then execution control passes to the next module in

6   sequence, namely the Determine Destination Application Availability module.

7

8   If, on the other hand, there is no link for the Working Client between the Source

9   Application and the Destination Application, the Destination Application's

10   Service Adapter searches the Destination Application's data store to find a

11   potential Matching Client for the Working Client using the following exemplary

12   search criteria: Social Security Number, Date of Birth, Last Name, and First

13   Name. This process is done to avoid duplicate entries of the Working Client in

14   the Destination Application's data store. Note: the Working Client may in fact

15   exist in the Destination Application's data store but a link may not exist for the

16   Working Client between the Source Application and the Destination Application.

17   If there are no potential Matching Clients, execution control passes to the next

18   module in sequence, namely the Determine Destination Application Availability

19   module.

20

LIBNY2/86357.3

1   If one or more potential Matching Clients are found, they are displayed to the

2   user. The user is then provided with three options.

3

4   Option 1: The user wishes to add the Person because none of the potential

5   Matching Clients actually matches the Working Client;

6

7   Option 2: The user desires to update and link the Person because at least one of

8   the potential Matching Clients actually matches the Working Client; or

9

10   Option 3: The user cancels and control is returned to the Dashboard.

11

12   Referring to options 1 and 2, if the user selects option 1, execution control passes

13   to the next module in sequence. If the user selects option 2, a determination is

14   made as to whether a link for the Working Client between the Source Application

15   and the Destination Application exists in the Integration Engine's Database.

16

17   Outcome 1 – No link For Working Client Exists in the Integration Engine's

18   Database

19

20   If there are no links for the Working Client in the Integration Engines database, a

21   determination is then made as to whether a link for the selected Matching Client

1   between the Source Application and the Destination Application exists in the

2   Integration Engine data store via the Verify Links module.

3

4   In response to the CheckLinks_RQ message, the Integration Engine's Service

5   Adapter will check the Links Table to determine whether a link for the Matching

6   Client exists between the Source Application and the Destination Application. A

7   CheckLinksResponse message is then constructed and transmitted indicating the

8   results of the query.

9

10   If a link for the selected Matching Client exists, an UpdateLinks message is

11   constructed and executed resulting in a link being created between the Working

12   Client in the source application and the selected Matching Client in the

13   Destination Application. Note: Since a link already exists, the PartyID of the link

14   for the person selected will be used when creating link. Thereafter, execution

15   control returns to the beginning of the iterative loop to process the next

16   Destination Application contained in the message envelope of the original

17   integration request message.

18

19   If, on the other hand a link for the selected Matching Client does not exist in the

20   Integration Engine data store, an UpdateLinks message is constructed and

21   executed, resulting in a link being created between the Working Client in the

1   Source application and the selected Matching Client in the Destination

2   Application. Note: Since no link existed for the Working Client or selected

3   Matching Client, a new PartyID will be created upon completion. Thereafter,

4   control returns to the beginning of the iterative loop to process the next

5   Destination Application contained in the message envelope of the original

6   integration request message.

7

8   Outcome 2 – Link for the Working Client Exists

9

10   If a link for the Working Client exists in the Integration Engine data store, an

11   UpdateLinks message is created and executed resulting in a link being created

12   between the Working Client in the Source Application and the selected Matching

13   Client in the Destination Application. Note: Since a link already exists, the

14   PartyID of the link for the working client will be used when creating link.

15   Thereafter, execution control returns to the beginning of the iterative loop to

16   process the next Destination Application contained in the message envelope of the

17   original integration request message.

18

19   After all Destination Applications contained in the message envelope of the

20   original integration request message having been processed, execution control

21   passes to the next module in sequence.

1

2

3   Determine Destination Application Availability

4

5   Referring to 9, upon completion of the preceding modules, the Verify Destination

6   Application Availability module executes in accordance with the following

7   exemplary process flow.

8

9   First, a SyncPerson_RQ message is prepared based on the original integration

10   request message as follows: the Source Application of the SyncPerson_RQ

11   message is set to the Source Application of the original integration request

12   message and the Working ClientID of the original integration request message is

13   used as a parameter of the SyncPerson_RQ message.

14

15   For each Destination in the Destinations Section of the

16   SyncPerson_RQ message the following steps occur:

17

18   First, a determination is made as to whether the Destination Application's

19   Services are available by issuing a ServiceAvailable_RQ Message to the

20   Destination Application Service Adapter.

21

1   If the Destination Application's Services are available, the Destination

2   Application is added to the message envelope (that is, the Destinations section) of

3   the SyncPerson_RQ message. Thereafter, execution control returns to the

4   beginning of the loop to process the next Application requiring synchronization.

5

6   If the Destination Application's Services are not available, execution terminates

7   and, in one embodiment of the present solution, an error message is generated

8   indicating the same.

9

10   Finally, after all Destination Applications have been added to the message

11   envelope of the SyncPerson_RQ message, control passes to the next module in

12   sequence.

13

14   5. Synchronization

15

16   Referring to FIG. 10, after the Verify Destination Application Availability module

17   has executed, control passes to the Synchronization module, which executes in

18   accordance with the following exemplary process flow.

19

20   First, signatures are checked via the Verify Links module (see above).

21

1 Next, a GetPerson_RQ message is constructed based upon information derived

2 from the SyncPerson_RQ message as follows: the Destination Application of the

3 GetPerson_RQ message is set to the Source Application of the SyncPerson_RQ

4 message and the PersonID parameter of the SyncPerson_RQ message is passed as

5 a parameter of the GetPerson_RQ message.

6

7 Next, the GetPerson_RQ message is sent to the Service Adapter of the

8 Destination Application. The Destination Application's Service Adapter retrieves

9 the appropriate customer demographic data, constructs and transmits a reply

10 message (GetClientResponse message) having the requested demographic data.

11

12 Next, a GetLinks_RQ message is constructed based upon information derived

13 from the GetClientResponse message. The GetLinks_RQ message retrieves other

14 relationships linked to the desired person, such as mother, father, son, etc.

15

16 Next, the GetLinks_RQ message is sent to the Integration Engine's Service

17 Adapter. The Integration Engine's Service Adapter retrieves any existing linked

18 relationships, constructs and transmits a GetLinksResponse reply message having

19 the linked relationships

20

1   Next, an UpdateLinksRequest request message is constructed using information

2   derived from the original integration request message. (Note: at this stage in the

3   process, the SynchPerson_RQ message is still being prepared for execution.)

4

5   Next, for each Destination Application in the Destinations section of the

6   SyncPerson_RQ message, the following steps occur:

7

8   Next, the SyncPerson_RQ message is further populated with the following

9   information: the Destination Application is loaded in the Destination section of

10   the SyncPerson_RQ message, the payload of the GetClientResponse message

11   (having demographic information) is loaded in the Payload section of the

12   SyncPerson_RQ message and the Links section of the GetLinksResponse

13   message is loaded in the Links section of the SyncPerson_RQ message

14

15   Next, the SyncPerson_RQ message is sent to the Destination Application's Service

16   Adapters. In response to the SyncPerson_RQ request, the Destination

17   Application's Service Adapter retrieves the desired data from the data store of the

18   Destination Application (sync data from the Source Application's data store to the

19   Destaintion Application's data store), constructs and sends a

20   GetSyncPersonResponse reply message indicating if the synchronization was

21   successful or not. If an error was encountered during the processing, this error

1 will be included in the message along with information about the error itself, such

2 as, number description, etc.

3

4 Next, the Link section of the UpdateLinks_RQ message is updated to include the

5 link information of the Links section of the GetSyncPersonResponse reply

6 message.

7

8 Control returns to the beginning of the loop to process the next Destination

9 Application in the Destinations section of the SyncPerson_RQ request message.

10

11 After all Destination Applications have been processed (synched), the

12 UpdateLinksRequest message is sent to the Service Adapter of the Integration

13 Engine. In response to the UpdateLinks_RQ request, the Integration Engine's

14 Service Adapter uses the information stored in the Links section of the

15 UpdateLinks_RQ to update the Links Table. More particularly, once

16 synchronization is complete, the SyncPerson_RQ message will add or update the

17 links for the Source and Destination(s) if necessary.

18

1   Thereafter, the Integration Engine's Service Adapter constructs and dispatches an

2   UpdateLinksResponse reply message indicating whether the process was

3   successful or not. If not, an appropriate error message will be returned.

4

5   Next, an UpdateSignature message is constructed for all applications in the Source

6   and Destination sections of the SyncPerson_RQ message, and dispatched to the

7   Integration Engine's Service Adapter. The Service Adapter then adds the date of

8   the synchronization to the Links Table.

9

10   Finally, in response to the UpdateSignature message, the Integration Engine's

11   Service Adapter constructs and dispatches an Output message based on

12   information derived from the SyncPerson_RQ and UpdateLinks_RQ messages.

13

14

15   ALTERNATIVE APPLICATION INTEGRATION FLOWS

16

17   FIGS. 12-16 depict alternative integration flows associated with certain aspects of

18   the present invention.

19

LIBNY2/86357.3

1 In each figure, the integrated software architecture is divided into levels, namely

2 several Application levels, a User Interface/Dashboard level, an Integration

3 Services Level and a Data Source Level.

4

5 As shown, the Application level contain standard native applications, namely,

6 Application A, Application B as well as Other Applications. The User

7 Interface/Dashboard Level contain the user interface. The Integration Services

8 level contain the requisite integration components, such as the integration engine

9 and service adapters. Finally, the Data Source Level contains the various data

10 stores that are created, managed and stored for the purposes of synchronizing data

11 across applications.

12

13 Referring back to the figures, FIG. 12 depicts how a user, using a standard

14 application (in this case, Application A), creates a client record for a new

15 person/client/customer/prospect in accordance with the present invention. As

16 evidenced by the illustrated flow, there is no interaction with any of the

17 integration components of the system, that is the Dashboard or the Integration

18 Services, during this process. Because of this, the integration components are

19 unaware of the new client record. This situation would be accounted for in future

20 work-flows by either the user or by the integration software.

21

1   Alternatively, FIG. 13 depicts how a user, using the Dashboard, creates a client

2   record for a new person/client/customer/prospect in accordance with the present

3   invention.

4

5   FIG. 14 depicts how a user, selects a Working Client using the Dashboard, in

6   accordance with the present invention. This integration flow assumes that the

7   selected Working Client already exists in the Link Table.

8

9   The flow of FIG. 15 depicts synchronization of a linked client and FIG. 16 depicts

10   how links between applications are established.

11

12

13   PRACTICAL APPLICATIONS

14

15   The following sections provide practical applications of the present solution in

16   order to fully appreciate features of the present solution.

17

18   In the use cases that follow, Agent Ms. Angie Baker will begin her workflow by

19   prospecting for a potential customer, a Mr. William R. Brown. Among the

20   various integrated applications on Ms. Baker's Dashboard include: a prospecting,

21   a discovery, an analysis application, an asset allocation application, a product

1   illustrations application and an electronic assistant application. For simplicity, the

2   foregoing applications shall hereinafter be referred to as a CDS application, a DIS

3   application, a PAS application, a PLAM application, an ISP application and EA

4   application, respectively. Ms. Baker's Dashboard also includes one or two

5   external, non-integrated applications, for example, a web browser application

6   such as Microsoft® Internet Explorer®.

7

8

9   Use Case 1: Creating a New Person and Pushing Information Into a Second

10   Application

11

12   This use case can take place over a period of several days or weeks.   After

13   prospecting with the CDS application, Ms. Baker meets with a potential customer

14   or prospect, a Mr. William R. Brown, and gathers information about Mr. Brown

15   using the Discovery application. Ms. Baker further analyzes the prospects

16   information using the PAS application.

17

18   To propose insurance policies to the prospect, Ms. Baker moves on to ISP to

19   create illustrations of the products for the prospect. When the prospect chooses an

20   insurance policy and chooses to open an investment account, Ms. Baker uses the

1   PLAM application to capture required investor information. Finally, Ms. Baker

2   uses the EA application to submit the new business information.

3

4   After many days of calling Mr. Brown for a follow-up meeting, Ms. Baker finally

5   sets up an appointment with Mr. Brown. Ms. Baker meets with Mr. Brown and

6   gathers information about Mr. Brown and completes a paper-based Fact Finder on

7   Mr. Brown during the meeting. Ms. Baker returns to her office to enter the Fact

8   Finder data into the Discovery application.

9

10   To begin entering the Fact Finder data into the Discover application, Ms. Baker

11   clicks on the "Search" button of the Dashboard.

12

13   The search dialog appears and Ms. Baker enters the name search criteria, such as

14   "Brown" in the last name field and "William" in the first name field, selects the

15   Source applications she wants to search (e.g. CDS, ISP), and clicks on the

16   "Search" button. After Ms. Baker submits the search request, the results appear as

17   a list of William Brown's found in the selected applications.

18

19   Ms. Baker highlights the William Brown she wants to work with (from the CDS

20   application database in this case since she first entered Mr. Brown's information

21   in the CDS application) based on the information displayed (such as address and

1   Tax ID) and clicks on the "Working Client" button, thereby setting William R.

2   Brown as the current client. The "Working Client: William R. Brown (CDS:)" is

3   then set and indicated in the Status area of the Dashboard. Note: the term

4   "Working Client" refers to both customers and prospects.

5

6   Ms. Baker then clicks on "PAS/Discovery" button. A dialog box opens asking

7   "This action will create information about the Working Client in PAS/Discovery.

8   Do you want to continue?" Ms. Baker sees a "Yes" button, a "No" button, and a

9   check box titled "Don't ask anymore. Just do it." Ms. Baker has become familiar

10 with the Dashboard interface and checks the "Don't ask" check box. If William R.

11 Brown records were already in the PAS/Discovery applications, the dialog box

12 would have said "This action will update existing information about the working

13 client in PAS/Discovery. Do you want to continue?" The existence of William R.

14 Brown in the PAS/Discovery applications is based on information in the Link

15 Table and therefore a search of the PAS/Discovery database(s) is not done.

16

17 The Integration Engine pulls William R. Brown's information from the CDS

18 application's database and pushes it into the PAS application's database and then

19 launches the Discovery application to displays the newly created William R.

20 Brown record in the Discovery application. Ms. Baker then enters the

LIBNY2/86357.3

1 information from the Fact Finder into the Discovery application for use in the

2 PAS application.

3

4 When Ms. Baker is done entering the information from the Fact Finder into the

5 Discovery application and because it is the end of the day on Friday, Ms. Baker

6 shuts down her computer and heads home for the weekend.

7

8

9 Use Case 2: Pushing Person Information From One Source Application Into

10 Another Destination Application

11

12 On the following Monday, Ms. Baker is scheduled to meet with Mr. Brown for an

13 implementation meeting. During this meeting she will use the ISP application.

14 To save time and eliminate re-keying of data, client specific information from the

15 CDS application will be pushed into the ISP application.

16

17 In preparation for the meeting with Mr. Brown, Ms. Baker turns on her computer,

18 which automatically launches the Dashboard. The Working Client is set to Mr.

19 Brown and the Working/Source Application is still set to the CDS Application as

20 evidenced by the caption "Working Client: William R. Brown (CDS:)" in the

21 Status area of the Dashboard.

1

2   Ms. Baker then clicks on the "ISP" application button on the Dashboard to push

3   information about Mr. Brown from the CDS application into the ISP application.

4   Note: since Ms. Baker previously checked the "Don't ask" check box, no dialog

5   box appears. Thereafter, the ISP application automatically launches and displays

6   Mr. Brown's record.

7

8

9   Use Case 3: Pushing Client Specific Information From One Source Application

10   Into More Than One Destination Application

11

12   The next day, Ms. Baker must use both the PLAM and the EA applications

13   because she successfully sold an insurance policy and an investment account to

14   Mr. Brown. Rather than following an application-by-application workflow, she

15   decides (as a sophisticated user) to populate these applications at the same time

16   with Mr. Brown's information.

17

18   Hence, Ms. Baker turns on her computer, which automatically launches the

19   Dashboard. Again, the Working Client is set to Mr. Brown and the

20   Working/Source Application is set to the CDS Application as evidenced by the

1   caption "Working Client: William R. Brown (CDS)" in the Status area of the

2   Dashboard.

3

4   Using the Dashboard, Ms. Baker clicks on the "Search" button, launching the

5   Search applet. The applet forms are automatically populated with information

6   from the Working Client. Ms. Baker selects the PLAM and EA applications and

7   then clicks on the "Synchronize" button to push Mr. Brown's information from

8   the CDS application to the PLAM and EA applications

9

10   After synchronization is complete, Ms. Baker clicks on the "EA" button, which

11   launches the EA application and displays Mr. Brown's record created in the EA

12   application as a result of the "Synchronize" action. After completing the EA

13   application, Ms. Baker launches the PLAM application through the Dashboard in

14   the same manner.

15

16

17   Use Case 4: Searching For A Person

18

19   After taking an extended leave of absence from work, Ms. Baker cannot recall

20   who she previously worked on and entered into the various applications.

1   Furthermore, during her leave her assistant Ms. Green worked on several cases,

2   which Ms. Baker is unaware of.

3

4   To get up to speed, Ms. Baker turns on her computer, which automatically

5   launches the Dashboard. However, the Working Client is no longer set to Mr.

6   Brown but to a different client.

7

8   When Ms. Baker receives calls from unfamiliar people, she clicks on the "Search"

9   button on the Dashboard and uses the name search function. She can then launch

10   into each application from the list of people displayed in the results area of the

11   search window by double-clicking on a person's row or by highlighting the person

12   and clicking on the "Launch Application" button. Using this method, she can get

13   background information on each person she has searched.

14

15

16   Use Case 5: Synchronizing Information

17

18   Later that day, Ms. Baker realizes that while she had linked Mr. William R.

19   Brown together, his address information is not the same across all of the

20   applications.

21

1   Thus, Ms. Baker brings up all occurrences of William R. Brown using the

2   Dashboard "Search" button. All of the same William Brown rows are

3   highlighted. She realizes that she missed an occurrence of William R. Brown in

4   another application. She highlights that row as well and then clicks on the

5   "Synchronize" button.

6

7   Ms. Baker confirms that the freshest information is in the CDS application. She

8   knows that the currently highlighted row in the results list will be used as the

9   source of the freshest information. By default, the currently highlighted row is the

10   "Working Client" row. She recognizes the currently highlighted row because it is

11   highlighted differently from the others. The other occurrences of Mr. Brown will

12   be updated with the information from the CDS application.

13

14   After synchronization, the search result list is refreshed with the updated

15   information.

16

17   Finally, FIGS. 17-22 are representations of user interface screens depicting

18   aspects of the present invention described hereinabove.

19

20   CONCLUSION

21

1   Having now described a preferred embodiment of the invention, it should be

2   apparent to those skilled in the art that the foregoing is illustrative only and not

3   limiting, having been presented by way of example only. All the features

4   disclosed in this specification (including any accompanying claims, abstract, and

5   drawings) may be replaced by alternative features serving the same purpose, and

6   equivalents or similar purpose, unless expressly stated otherwise. Therefore,

7   numerous other embodiments of the modifications thereof are contemplated as

8   falling within the scope of the present invention as defined by the appended

9   claims and equivalents thereto.

10

11   Moreover, the techniques may be implemented in hardware or software, or a

12   combination of the two. Preferably, the techniques are implemented in computer

13   programs executing on programmable computers that each include a processor, a

14   storage medium readable by the processor (including volatile and non-volatile

15   memory and/or storage elements), at least one input device and one or more

16   output devices. Program code is applied to data entered using the input device to

17   perform the functions described and to generate output information. The output

18   information is applied to one or more output devices.

19

20

1   Each program is preferably implemented in a high level procedural or object

2   oriented programming language to communicate with a computer system,

3   however, the programs can be implemented in assembly or machine language, if

4   desired. In any case, the language may be a compiled or interpreted language.

5

6   Each such computer program is preferably stored on a storage medium or device

7   (e.g., CD-ROM, hard disk or magnetic diskette) that is readable by a general or

8   special purpose programmable computer for configuring and operating the

9   computer when the storage medium or device is read by the computer to perform

10   the procedures described in this document. The system may also be considered to

11   be implemented as a computer-readable storage medium, configured with a

12   computer program, where the storage medium so configured causes a computer to

13   operate in a specific and predefined manner.

14

15